Multimedia Tools and Applications (2021) 80:21101–21133
https://doi.org/10.1007/s11042-021-10759-6

Paving the way to collaborative context-aware mobile applications: a case study on preventing worsening of allergy symptoms

Pablo Caballero[1] · Guadalupe Ortiz[1] · Alfonso Garcia-de-Prado[1] · Juan Boubeta-Puig[1]

Received: 31 July 2019 / Revised: 21 January 2021 / Accepted: 25 February 2021 /
Published online: 12 March 2021

Abstract
In recent years, the evolution of smartphones and their software applications has grown exponentially; together with the advance of the Internet of Things and smart cities, it has raised huge demand for services and applications in these domains. Although the wide range of mobile applications is unquestionable, citizens already demand that applications adapt to their specific needs and situations in real time, that is, that they are context-aware. However, context-aware mobile applications are often very limited and miss out on the opportunity of benefiting from feedback provided by citizen collaboration. In order to fill this gap, this paper proposes a context-aware and collaborative software architecture and mobile application. In particular, we have implemented them in the scope of e-health, more specifically in the area of seasonal allergies, which cause allergic people to experience annoying symptoms that could be avoided by having access to pollen information in real time. Furthermore, they will also benefit from citizen collaboration through the knowledge of the symptoms other allergic people with the same allergy and in the same location are experiencing. To this end, users will be able to provide their symptoms at any time through their mobile application and the proposed architecture will constantly process that information in real time, sending notifications to users as soon as reported symptoms are seen to exceed a certain threshold. The architecture's performance, the application's resource consumption and a satisfaction survey of the app's usability and usefulness have been tested; all results have been fully satisfactory.

Keywords Collaborative context awareness · Mobile application · Contextual alert · Internet of things · Smart city · E-health

✉ Guadalupe Ortiz
guadalupe.ortiz@uca.es

Extended author information available on the last page of the article

1 Introduction

Over the last few years, the number and use of smartphones has grown exponentially worldwide, and in many countries the number of Internet connections made from a smartphone is already larger than from desktop or laptop computers [47]. For these reasons, the development of mobile applications has become a niche market for software companies, as well as a research topic of great interest. The success of mobile devices in recent years undoubtedly lies in the advances that have been achieved in terms of improving their hardware capabilities, reducing weight, lowering the costs of mobile communications and faster Internet connection, among others. But without a doubt, the evolution and versatility of mobile software applications (apps) have also boosted their usage and greatly contributed to their success.

There is no doubt that this development has been parallel to that of the Internet of Things (IoT), which has clearly favoured the provision of information by multiple sensors and other devices connected to the Internet in recent years [30]. The IoT servers and platforms capture this data and facilitate its acquisition and consumption by other computer applications, mobile or not. The amount of generated data is enormous and the term Big Data was coined, referring to the large amount of heterogeneous data that flows along the information systems and which is analysed with the aim of improving decision-making in the domain in question. It is in this technological context that smart cities emerge, making use of the data generated by the IoT, local administrations, public companies and end users to provide citizens with a better experience of life in their city [77].

On the other hand, context awareness has become a fundamental requirement in multiple software applications [67]. In computing, context can be defined as information that can be used to characterize the situation of a person, place or object that is considered relevant for the interaction between a user and a software application, including the user and the applications themselves [19]. A system is considered context-aware when it uses context to provide relevant or tailored information and/or services to the user [1], where relevance depends on the application domain. Indeed, context-aware software solutions have greatly increased in popularity and are highly demanded, especially by mobile users, as today's users expect greater adaptation of the environment, and therefore of the software they use, to their personal needs at each location and in each situation. Some immediate examples of widely established context awareness are users' mobile device warning them if there is traffic congestion on their way to work, windshield wipers turning on automatically if it rains, or the mobile device warning them if there is a friend nearby. However, there are a limited number of context-aware services and applications from which users can benefit; there are still many services and applications that do not benefit from this feature, no doubt due to the difficulties of designing, implementing and maintaining a context-aware software infrastructure, not only to receive context information, but also to make use of it in order to provide advantageous services that can be customized according to user needs. Not surprisingly, the European Union identifies, among the Horizon 2020 challenges, research and development for context-aware IoT computation [24] and smart cities and e-health in the preliminary documents for Horizon2030 [23]. Besides, concerning the matter of sharing information across platforms, collaborative architectures for data sharing in the scope of the IoT and smart cities are an essential requirement de facto for giving additional value to any decision-making process [11].

When considering the implementation of collaborative context awareness in any field, and especially in the field of the IoT and smart cities, we have to bear in mind that such large

amounts of generated data require a software architecture that allows fast data processing, preferably in real time. Besides, the architecture has to permit discarding, without additional resource consumption, those data that are not relevant to the user in question. New technologies, such as Complex Event Processing (CEP), have risen in order to provide this constant data processing in streaming [45]; with CEP, we can analyse and correlate huge amounts of data which flow through information systems in real time and with this aim we proposed several software architectures in the past [27–29]. In this paper, we will improve our previously proposed CARED-SOA architecture so that we benefit from obtaining context from app users' collaboration. Therefore, the main aim of this paper is to propose a software architecture and app that permits obtaining *context* information from mobile app users' *collaboration* which feeds back to the software architecture's server side and reverts to benefits for the app users themselves.

In particular, we have focused on an Android e-health application that is increasingly in demand in society: alerting those interested in pollen concentration levels, generally people allergic to pollen. Although there are some applications for this purpose, they do not tackle end user's own characteristics (such as their most common symptoms, which may not necessarily be the same for all patients) and other contextual characteristics (such as the symptoms that other allergic people in the same location are experiencing in real time since the symptoms may vary depending on the season or even on the day). Therefore, we have provided citizens with a collaborative and context-aware application (AlergiApp) that improves the quality of life of people allergic to pollen in smart cities. Besides, the system stores the historical data of all reported symptoms associated to the allergen type to which the reporting user is allergic. This data can be very useful for medical specialists not only for research purposes, but also to support their patients.

The rest of the paper is organized as follows. Section 2 describes related work on the matter of architectures for collaborative context-aware apps as well as on allergy information and notification apps. Then, Section 3 presents a motivating scenario and the case study to be used along the paper and Section 4 introduces the technological background in order to facilitate the understanding of the proposal. Afterwards, the high-level architecture description, the collaborative context-aware solution and the technical implementation for collaborative context-aware real-time notifications are presented in Sections 5, 6 and 7, respectively. Experimental results concerning performance, resource consumption and user satisfaction are explained in Section 8. Finally, Section 9 outlines conclusions and future work.

2 Related work

Context, as previously introduced, refers to any information that can be obtained related to a person, place or application that might be relevant in order to improve the interaction between the final user and the application [19]. Context information is in general specific to each system, so that some information might be considered as part of the context in a given system but not in another one. This is why context has been faced from several perspectives in a wide range of particular proposals and classifications [17, 35, 42, 56, 67, 75]. In particular, context awareness supports the fact that the obtained context information is properly used by the system so as to improve the mentioned user-application interaction [1], adapting the system's behaviour to the particular needs of a specific user. Even though there is still a number of gaps to be covered for context-aware applications in general [34] and for context awareness for the

IoT in particular [30], some relevant approaches are examined in this section. Particularly, we first highlight the most relevant frameworks and middlewares proposed in recent years in terms of context awareness. Secondly, we focus our attention on context-aware recommendation systems, now for mobile apps, as well as on proposals related to context-aware team collaboration. Finally, we focus on mobile applications for context-aware e-health in general, and on those specific to allergy control in particular.

2.1 Relevant frameworks and middlewares for context awareness

Some prominent frameworks and middlewares for context awareness have been proposed over several years: CoWSAMI is a middleware which gives support to context, supplied by Athanasopoulos et al. [8]. It provides a Context Manager in order to deal with context sources. They provide services as interfaces to collect information. CAS-Mine is a framework presented by Baralis et al. [9], which focuses on discovering relevant relationships between user context data and invoked services through a high-level abstraction of both user habits and service characteristics, depending on the context. However, it does not provide an adaptation to context mechanism. The paper from Li et al. [44] is also of interest: they provide a framework for context provisioning, which can include new context provisioning schemas dynamically. They also facilitate the definition of new context schemas for additional application domains. Yu et al. present PerCAS, a model-driven approach which enables web service adaptation to user personal context preferences or user personalization [76]. The paper from Gilman et al. [31] deserves special attention. They provide a framework for context-aware pervasive services through a complex architecture composed of several components, among them a context reasoner, context discoverer and observers, handlers and managers. Their framework implies that context providers and actuators are designed specifically for their low-level context ontology and suffer from communication delays.

Although we consider that the frameworks described in this subsection have been among the most relevant in terms of context awareness in the last 10 years, and they are worth mentioning, they are not comparable to the proposal described here since they are mainly focused on context discovery and service adaptation, while our proposal is based on citizen collaboration in terms of providing specific context of the application domain and the customization of notifications to the user in question depending on his/her context. For these reasons, we will further define the related work throughout the following subsections.

2.2 Context-aware recommendation system for mobile apps and team collaboration

Regarding context-aware recommendation systems for mobile apps, we find several approaches: Pessemier et al. propose a recommendation system for mobile news which learns from user tastes and user feedback during the evaluation period [18]. Alhamid et al. provide a recommender system based on user preferences in social networks and the user's biological signals [4]. All these middlewares and proposals lack facilitating user collaboration to benefit other users. In the case of Zavala et al. [78], they use a machine learning algorithm to infer the context and use semantic web technologies to model it; however, they do not allow the explicit collaboration of users by providing their context, nor do they provide any particular app which benefits from it.

Concerning team collaboration, the ESCAPE framework and the inContext project deserve a special mention. Both are proposed by Truong et al., in [70, 71], respectively, and deal with a

service-based context-management system for team collaboration. Even though the concepts provided for exchanging and making use of context information in a service-based scope are high-quality ones, the narrow scope of their main focus (collaboration teams) weakens the value of this proposal for the scope of this paper. In more recent years, we have found additional proposals on mobile collaborative systems: in 2014 Benitez-Guerrero et al. presented a proposal of interest on contextual data sources to provide domain-specific information to the user [12]; however, there is no chance for the user to collaborate proactively, the system obtaining some contextual information from the mobile device. Besides, they perform an offline analysis of the data, while we provide real-time processing and notification. Similarly, Montané-Jiménez et al. present a context-aware framework for improving collaboration between videogame users [48], where the context obtained is the player's performance in a particular videogame. Chung abstractly sketches a mobile collaborative system with the aim of facilitating decision-making between several participants in an organization [16]; context is not taken into account by this proposal in its early stages. Similarly, Botev et al. propose CollaTrEx framework [14] for in-situ collaboration within groups of learners performing educational activities where spatio-temporal context is used for determining the available activities. Please bear in mind that in these proposals the aim is to facilitate collaboration between users within the same organization, but not to improve context knowledge by citizens' proactive collaboration.

At this point, we can summarize that context awareness has been a topic of great interest throughout the last few years; with the advance of mobile devices and the lower cost of mobile communications it has also aroused considerable interest in the development of context-aware apps. However, although we find proposals related to context awareness in mobile development, we observe that (1) they do not facilitate citizen collaboration so as to improve context awareness and services to other citizens; (2) few approaches propose real-time processing and (3) in no case do we find an interest in the evaluation of resource consumption and response time for the mobile user (since this study is not carried out in any of the previous proposals for mobile apps), which is key to users' acceptance of an app.

2.3 Context-aware mobile apps for e-health and allergy control

In relation to context-aware mobile apps for e-health in general, there are some incipient proposals; for instance Rahman et al. present a framework which selects suitable e-health services according to the user context obtained through body sensor measurements [58]. Additionally, Kim et al. also provide context-aware recommendations for selecting suitable e-health services [41]. Also of great interest is the proposal by Casino et al. [15], where they propose an architecture, illustrated with a case study of route recommendation, of a health-related context-aware recommendation system. Although they emphasize the importance of citizens' collaboration with their personal data to improve e-health applications, the proposal only considers their collaboration with static data (their age, diseases, etc.) and does not consider the collaboration of citizens with dynamic data in real time, which is the differentiating feature of our proposal. These proposals only take into account the information from the user in question and do not benefit from other users' feedback and context information which would definitely enrich the service provided to users.

Concerning existent mobile applications specific to allergy control we can find several; however, none of them provides context-aware notifications that also benefit from citizen collaboration. We have selected the apps that offer more functionalities and are updated. The

most basic ones that can be mentioned are the Accu Pollen Allergy Tracker [65], which only provides information on pollen counts; AlertaPolen [43], Allergy Pollen Count [20] and *Polen REA* [37], which together with the information on pollen counts provide alerts on the specific pollen types selected by the user; Pollen News [3], which provides information and predictions on pollen counts; and Melbourne Pollen Count [72] which provides information and predictions on pollen counts together with the selected alerts. Among the apps offering more functionalities we can mention, for instance, *Polen Control* [5]: it is an application with a user-friendly design, which allows both checking pollen levels in the user's location and reporting user symptoms with the aim of having a history that can be later consulted. Although the app shows the pollen levels, it does not offer the possibility of obtaining information about a specific type of pollen; moreover, although they indicate that the application can receive pollen alerts, we have not really been able to verify that this functionality actually worked, since we did not receive any alert. We can also highlight PollenWise [57], which provides information about pollen concentration as well as predictions, and permits saving information about how you feel and based on such information it can make suggestions about when would be the best time for doing physical activities outdoors. Additionally, Sensio Air [74] also provides information about pollen concentration as well as predictions and permits saving how you feel with the aim of offering you historic information about what caused you to feel allergy symptoms. Finally, the most relevant app we have found is *Pollen* [61], an app made collaboratively among several countries with a simple and user-friendly interface, which provides enough information about each type of pollen and their concentrations. It also provides pollen count predictions as well as alerts on those pollens selected by the user as well as storing information about how the user has felt with the aim of being able to review a history on how they have been feeling over time. However, in this application the user does not benefit from the information reported by other citizens, since this information is only provided to experts for further analysis.

As a conclusion, the most relevant difference of all these applications with the one proposed in this paper is that none of them uses the symptoms reported by the user in order to have a prognosis of the level of allergy and its symptomatology as well as to help other users in the same location and with the same allergies prevent such symptoms. In addition, none of them take into account the user's physical activity to personalize the alerts in each moment. This ability to provide more contextualized forecasts, thanks to the collaboration of other users, and the real-time processing of such data has not been found in any other app and is definitely what characterizes our proposal.

To end with, Table 1 summarizes the comparison conducted throughout the Related Work section.

3 Motivating scenario and case study

According to the European Academy of Allergy and Clinical Immunology, there are around 150 million allergic people in Europe [22]; 50 million in the US according to the Asthma and Allergy Foundation of America [7]. Despite the high occurrence of such allergies, every spring allergy sufferers are surprised by the effects they have on their bodies.

It is no coincidence that the spring months are becoming harder for the people affected by allergies. According to the World Allergy Organization there is an increase of allergic diseases worldwide; pollution and climate change only aggravate the symptoms caused by the body's

Table 1 Summary of comparison with related work

Topic	References	Discussion
Relevant frameworks and middlewares for context awareness	[8, 9, 31, 44, 76]	The most relevant frameworks and middlewares in terms of context awareness focus mainly on context discovery and service adaptation, not taking into account the citizen's collaboration in terms of providing a specific context of the application domain and the personalization of notifications to the user in question according to their context. The democratization of data and citizen collaboration has emerged in recent years and has yet to be integrated with the existing work in the matter of context awareness.
Context-aware recommendation systems for mobile apps	[4, 18, 78]	In the case of recommendation systems for apps, although we find proposals related to context awareness, we observe that the recommendations are mainly based on inferences from static contexts or contexts of the user's environment, but do not benefit from the feedback and contribution that the user himself/herself can make in the application domain, and above all from the processing of that context in real time.
Apps for team collaboration	[12, 14, 16, 48, 70, 71]	The proposals that we find in which collaboration is facilitated in a mobile app focus on facilitating collaboration between users within the same organization, but not with the aim of improving knowledge of the context through proactive collaboration of citizens, but in order to perform a task among all users in a collaborative manner, which is not the purpose of this paper.
Context-aware apps for e-health	[15, 41, 58]	Although there are more and more proposals of apps for e-health and in particular context-aware e-health apps, the proposals we find only take into account the information from the user in question and do not benefit from feedback and contextual information from other users, which would certainly enrich the service provided to users.
Apps for allergy control	[3, 5, 20, 37, 43, 57, 61, 65, 72, 74]	Regarding the apps for controlling pollen allergy, we see that there are several proposals covering the basic functionalities: they report on pollen concentration levels, and alert users when these exceed a certain threshold. Some provide predictions of concentration levels and also some more advanced ones allow users to report on how they feel and/or their symptoms in order to store these details and be able to consult their history later. The most relevant difference of all these applications we propose is that none of them uses the symptoms reported by other users to have a contextualized prognosis of the level of allergy and its symptomatology as well as to help prevent those symptoms in other users in the same location and with an allergy to the same pollens. In addition, none of them take into account the user's physical activity to personalize the alerts in each moment.

immune reactions. The risk of suffering a serious reaction is therefore increasing and, even though traditional methods of treatment are still effective, it would be desirable for allergy

sufferers to be aware of pollen concentrations and current symptoms in real time, so they can act quickly and prevent greater evils.

In this sense, with current available technologies, continuing to act reactively to combat allergies is a setback to their life quality. By making use of the information provided by users on pollen levels at a given location in real time, a feedback system could be implemented which can detect and warn about the appearance of certain pollen concentrations and their possible consequences.

This way, not only could life quality be improved, but also people could better understand what allergies are and why they occur: what symptoms they cause, how frequent they are, and the threshold from which those symptoms appear.

In this scope, an ideal scenario would be to provide allergy sufferers with an application which can show personalized information about pollen. We have to bear in mind that the same pollen concentrations do not always cause allergy sufferers the same symptoms, as we need to factor in each person's other atmospheric, physiological and circumstantial conditions [6]. In order to be able to show that information, not only do we need to know the state of pollen concentrations at any given time in a given place, but also which main symptoms those concentrations of that particular pollen are causing, which symptoms other people affected by pollen allergy are currently experiencing and, finally, which symptoms usually affect the user in question. With all this information we can give more detailed, personalized and useful information to application users.

Besides, as a result of fruitful discussions with the pulmonologist, the app is to remain as simple and intuitive as possible, to achieve a greater adherence of people/ with low technological skills and/or older people whose reaction to seasonal allergies is aggravated by its coexistence with other respiratory diseases such as Chronic Obstructive Pulmonary Disease (COPD). Since many of these patients also need to exercise to maintain their lung capacity, it would be advisable to include some exercise-related functionality that allows them to adjust their alert levels when they are going to do physical activity, decreasing the risk of an allergy attack at that time. Likewise, it is considered of interest that the pulmonologists could know, based on real data, which symptoms are frequently being detected with the different allergies in a certain geographical area, in order to be able to advise and guide the patients who come to the consultation.

Taking all the above considerations into account, such an application should provide the following functionalities:

1. Showing information about pollen levels: users should be able to check information about pollen levels in their own location.
2. Allowing users to collaborate by reporting their symptoms into the system: if we want a retro-fed system, we need to obtain information from the users through our application, of course with users' consent according to the general data protection regulation in force. This information will allow us to know how each pollen type affects users in a given location and be able to inform them about the symptoms that are occurring at this time thanks to the collaboration of all app users. Please note that allergy symptoms do not always occur exactly at the moment the allergen is inhaled, but are given or aggravated by an accumulation of inhalation over time; therefore it is possible for the citizen to inhale a lot of pollen on the way to their children's school and report the symptoms an hour later at work, even if they are at the other side of city. Given the fact that most cities do not have

more than one pollen counting station, when we mention the locations in which symptoms are reported and notified, we refer to entire cities.

3. Receiving alerts: the system is useless if we do not have a way to warn users when pollen levels are rising as well as when other users are experiencing common symptoms. Through mobile notifications, we can provide the user with this information quickly and effectively. The alerts will be contextualized given the user's allergies and location, as well as the physical activity he/she is currently doing, together with other users' symptoms in the same location with the same type of allergy.
4. Allowing configuration according to the user: obviously, not all people are allergic to the same pollen types nor do they react to them the same way. The application must allow the user to decide what information he/she wants to be warned about.
5. Showing data in a simple, intuitive and efficient way: when it comes to mobile applications, it is essential to achieve an attractive and functional design and reasonable resource consumption. An application can fulfil all the required functionalities, but an unattractive or unintuitive design, or excessive resource consumption in the device, will cause an immediate rejection by the user.

For this application to be able to offer all these functionalities we will provide a server-side architecture capable of (1) processing data from various sources in real time, (2) detecting events of interest depending on the context of each user, (3) facilitating citizen collaboration to provide useful feedback data and (4) sending personalized notifications in real time.

4 High-level architecture description

To meet the needs highlighted in the motivation, in this section we present the proposed architecture, which consists of three modules (see Fig. 1): firstly, the input module is in charge of gathering data from several sources from different domains that are later processed and sent to the mobile application. Secondly, the server-side module takes the input data and processes or stores them so that the application can consume them when necessary, according to the pattern of interest. Finally, the client-side module consists of the mobile application in charge of presenting the information to the user in a friendly and understandable way and will also feed back the server-side module. These three modules are explained in more detail in the following subsections.

4.1 Input data module

The input data module (see input module in Fig. 1 gathers heterogeneous data from a variety of sources; in our case study implementation we will collect two types of data coming from two types of source:

- Pollen data: this data sources will provide the system with the amount of each pollen type, updated as often as possible. Pollen data is obtained from several providers depending on the location. Besides, information about each type of pollen is collected, so that the user can know relevant aspects about them. Some of the data returned by this source are the scientific name, a description or the pollen calendar.

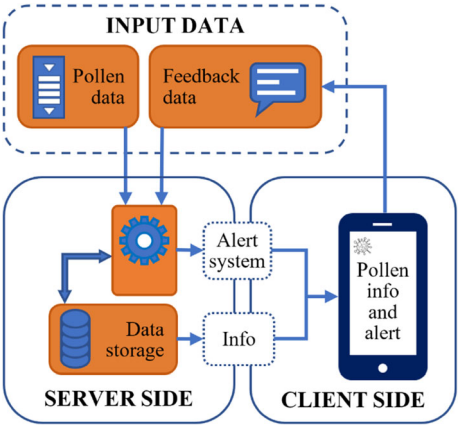


Fig. 1 High-level architecture description

- Feedback data: user feedback information must be able to re-enter the system. These data are obtained from the app and will be used to generate statistical information as well as alerts for frequent symptoms in a city.

4.2 Server-side module

This module is composed of several software elements interconnected between them and with the rest of the modules. There is a clear division between this module's two responsibilities:

- Generating statistical information that associates pollen types to symptoms and vice versa: the objective is to associate the symptoms reported by users with the possible types of pollen that may cause these symptoms. Thanks to this association, it will be possible to obtain, not only the association between symptoms and types of pollen causing them, but also the percentage of allergic people affected by such symptoms, in such a way that the final user will be able to have information such as, for example: "The olive tree causes dyspnoea to 80% of users allergic to this pollen type", or " Dyspnoea appears in 100% of users allergic to gramineae".
- Generating real-time alerts on common symptoms in a city: in this case, the system will be responsible for continuously analysing the information reported by users, with the aim of detecting the appearance of certain patterns that indicate that a particular symptom is appearing repeatedly over a short period of time in a particular location. This way, the system will be able to generate a contextual alert and send it to all subscribed users in such a location who are allergic to the type of pollen causing it. For example, if many users allergic to olive pollen report asthmatic symptoms in Cádiz, the system will generate a contextual alert for all users in Cádiz who are allergic to olive pollen indicating: "Many users in Cádiz are reporting asthma", so that the user can be prepared for the possible appearance of this symptom.

However, not all users in a city should receive such alerts, since, if the asthmatic symptom was not caused by the olive pollen type, users allergic to the olive tree would not be interested in

this information. Therefore, there is a connection between the system for generating alerts and the statistical information system, with the aim of obtaining, given a symptom, all possible types of pollen causing it, so that contextual alerts are only sent to users who are allergic to any of the relevant pollen types.

4.3 Client-side module

The client-side module consists of a mobile application. This will be connected to the server-side module to obtain the data to be displayed (see Fig. 2, arrow (a)) and to receive the notifications in real time (Fig. 2, arrow (d)). In order to receive such notifications the user should have chosen which pollen type alerts he/she wants to subscribe to (Fig. 2, arrow (b)). In addition, the app will submit the data introduced by the users back to the server-side system module (see Fig. 2, arrow (c)), generating feedback that will allow the app to offer context-aware and personalized information that has been described in the server-side module. Such feedback is taken into account when the server-side module broadcasts alerts to affected users where applicable, as previously mentioned (Fig. 2, arrow (d)).

5 Collaborative context-aware solution

This section describes the entire feedback process of the system, with special attention to the collection of information from users, its processing and the final result, which will provide relevant information to other users thanks to our collaborative context-aware proposal.

5.1 Contextual information

In this section we describe the various types of context taken into account in the proposed system:

Location First of all, we take into account the user's location to notify alerts only specific to the city in which he/she is located, whether this is selected statically by the user or obtained automatically by the mobile's GPS.

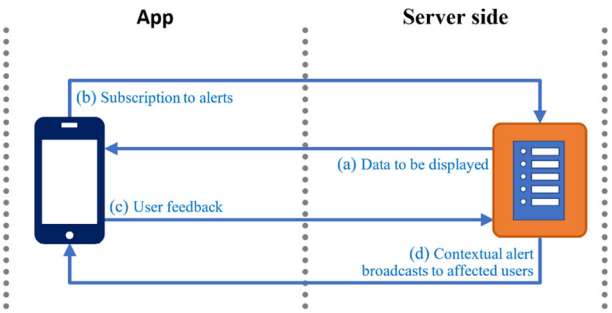


Fig. 2 Communications between the server side and the app

Symptoms of other users in the same location and with the same allergy Secondly, and especially noteworthy of this system, we take into account the symptoms of all users with the same allergies in the same city for contextualized and customized notifications about those symptoms.

Physical activity Thirdly, the user's physical activity is also taken into account. On the one hand, notification thresholds will be lower when the user is doing physical activities, since more air is inhaled and therefore more allergens. Thanks to the use of Context Awareness' Google API we can know if the user is running, cycling, walking, etc. The greater the physical effort, the lower the thresholds for notifications to be sent. In this sense, running and cycling will have the lowest thresholds, followed by walking and finally users at rest will receive notifications with higher thresholds. On the other hand, when the user reports a symptom, he/she will also report the physical activity that he/she was doing when experiencing it, so that the activity information is also stored together with the symptoms and allergies. This can be very useful to conduct statistics to be analysed by pulmonologists.

5.2 Collection of information from users

The first step which permits obtaining context information thanks to users' collaboration is that app users fill in data into the application. Specifically, they must report any symptom through the application screen intended for this purpose. In order for the reports to be anonymous, no personal user data is sent. A report consists only of the reported symptoms (since several symptoms can be indicated in the same report) and physical activity, the list of pollen types to which the user is allergic and the city in which the user is when reporting them. Bear in mind that the user is expected to record the pollen type to which he/she is allergic the first time they use the app, and such information is saved in the device. It is also important to mention that a single user who makes a report is already generating relevant information for other users, since it will contribute to updating the statistical information system and the detection of frequent symptoms in a given location.

5.3 Information processing

Once the user sends the report, it reaches the system server-side module, where it is used to update statistical information and to generate a frequent symptom alert when necessary. Next, the step-by-step process carried out for each of these actions is explained:

To illustrate how the statistics have been obtained, we provide a simple example below. Let us suppose that Table 2 represents the initial state of the statistical database; only relevant data is shown to facilitate the proposal understanding.

Let us now assume that any user allergic to olive tree, gramineae and banana reports dyspnoea and dermatitis. When the report reaches the system, firstly, the pollen level in the city in question is queried. If it is determined that there are no high levels of a pollen type to which the user is allergic, the information received is discarded. Suppose that, in this case, banana is discarded, but olive tree and gramineae are taken into account.

The association between reported symptoms and pollen types is then added to the previous database (see Table 3):

Table 2 Initial state of the database for illustration purposes

Symptom	Pollen type	Verified
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Rhinitis	Gramineae	1

The system will then check the database to look for any associations between reported pollen types and symptoms that may not have been reported. As a matter of fact, both olive tree and graminaea have two associated symptoms, conjunctivitis and rhinitis, that have not been reported by the user. For this reason, the system itself will add this association, setting the value of the *Verified* column to 0. After that, the status of the database would be as shown in Table 4.

At this point, we can obtain the desired statistical information:

- If we want to know the percentage of users who have conjunctivitis due to olive tree pollen, we will divide the verified associations for such a symptom and pollen type by the total number of associations (verified and unverified). There are 3 verified associations, while the total is 4, so 75% of users have conjunctivitis due to the olive tree pollen.
- If we want to know the percentage of users who have rhinitis because of gramineae, by following the same process we will obtain 50% of users.

In parallel, the CEP engine will receive the data and check if the established pattern is fulfilled: frequent symptoms must be detected, so a minimum number of events must be reported at a given time in a given location. Specifically, for testing purposes we will send a notification if 5 occurrences of the same symptom are reported in the same location over a period of 5 h. After such a period of time, the information managed by the CEP engine is discarded.

The application approaches collaboration from two points of view. On the one hand, it approaches it from the C-IoT model defined in [11], where the 3 levels, sensing, gateway and services, are established. In this sense, we sense data both from the data sources of allergen concentrations and the data provided by users about their symptoms. Our CEP engine serves as a gateway, which will allow us to aggregate and analyze these data in real time, thanks to which we will be able to offer a twofold service: the feedback provided to the user for his/her

Table 3 Association between reported symptoms and pollen types (added rows in italics)

Symptom	Pollen Type	Verified
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Rhinitis	Gramineae	1
Dyspnoea	*Olive tree*	*1*
Dermatitis	*Olive tree*	*1*
Dyspnoea	*Gramineae*	*1*
Dermatitis	*Gramineae*	*1*

Note that the *Verified* column indicates that the association between symptom and pollen type has been verified, since the symptom has been reported directly by a user allergic to the pollen type in question.

Table 4 Association between reported symptoms and pollen types' expected symptoms (added rows in italics)

Symptom	Pollen Type	Verified
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Conjunctivitis	Olive tree	1
Rhinitis	Gramineae	1
Dyspnoea	Olive tree	1
Dermatitis	Olive tree	1
Dyspnoea	Gramineae	1
Dermatitis	Gramineae	1
Conjunctivitis	*Olivo*	*0*
Rhinitis	*Gramineae*	*0*

own benefit in terms of his/her allergy control and the possibility for a specialized doctor —pulmonologists and allergy specialists— to study the history of allergen levels and their correlation with the symptoms, for research purposes and/or information and help to patients in his/her practice. On the other hand, the collaboration is also done through the democratization of data, where the citizen is sharing his/her symptoms to help and encourage an app that benefits the evolution towards a more intelligent city [2].

6 Technical implementation for real-time context-aware notifications

According to the high-level architecture description, the technical implementation for real-time context-aware notifications requires real-time processing. Due to the gathering of heterogeneous data from several sources and requiring such real-time processing and notification, we considered an Event-Driven Service Oriented Architecture (ED-SOA or SOA 2.0.), used in conjunction with a CEP engine, to be the best choice for implementation. Let us first introduce the technologies and afterwards explain how these were used to implement our proposal.

6.1 Event-driven service oriented architectures

A Service Oriented Architecture (SOA) consists of a paradigm for the design and implementation of loosely coupled distributed systems which make use of services for their implementation. These architectures easily integrate third-party systems in a flexible and loosely coupled way, so that the focus can remain on the business process rather than on the technologies. This way, system maintenance and evolution are facilitated when the system requires changes and costs are reduced [54]. Therefore, the service orientation concept is based on the idea of offering a well-defined interface which provides communications based on a standard protocol, where currently the provider and the consumer are completely decoupled. Decoupling is obtained thanks to technology independence.

One of the most common ways to implement SOAs are web services, and particularly REpresentational State Transfer (REST) services. These are self-descriptive software modules which can be accessed through a network and which develop a task facilitating machine to machine interoperability [54]. REST is an architectural style for distributed hypermedia systems where services provide resources identified by URLs [59], which emerged as an

alternative to more traditional SOAP web services. Communications between REST services and their clients take place using HTTP main operations.

Besides, with the growth of service components and processes in service oriented applications, a new service infrastructure is required for maintaining applications in a flexible way. These requirements are fulfilled by an Enterprise Service Bus (ESB), which supports well-known web service standards and provide support for a message middleware [54]. An ESB integrates services with complex architectures through a messaging system, supplying interoperability among diverse applications and components through standard interfaces, which allows applications to be offered as services in the ESB.

However, when there is a need for reacting to events, an Event-Driven Architecture (EDA) might be necessary: in particular, a SOA 2.0. evolves from traditional SOA. In SOA 2.0., communication between users, applications and services is carried out by events, rather than using remote procedure calls [46]. To facilitate this paradigm, a software abstraction layer is required to integrate diverse heterogeneous data sources and distributed invocations [55]. These functionalities are offered by the just explained ESB, which permits interoperability among several communication protocols and heterogeneous data sources and targets. In these types of architecture where large amounts of events are received and have to be processed, a message broker can be of great use. Message brokers implement an asynchronous mechanism which allows source and target messages to be completely decoupled, as well as permitting storing the messages in the broker until they can be processed by the target element. These brokers may use standard message queues or be combined with a publish/subscribe mechanism [10], where messages are published according to a set of topics and users subscribe to the topics of their interest.

Despite all the advantages provided by SOA 2.0, this type of architecture might not support the analysis and correlation of large amounts of data in terms of events in real time. To meet this requirement, it is necessary to integrate CEP [46] within SOA 2.0. CEP is a technology that permits capturing, analysing and correlating of a large amount of heterogeneous data — simple events — with the aim of detecting relevant situations in a particular domain [38]. Event patterns specify the conditions to be met (complex events) in order to detect such situations in the software capable of real-time data analysis (the CEP engine).

6.2 Technical development details

Once we have conceptually explained SOA 2.0 and CEP and motivated their use, we are going to explain the particular programming languages and platforms chosen for the architecture's development.

As previously explained, a SOA can be implemented through REST services. In a REST Application Programming Interface (API), HTTP GET, POST, PUT and DELETE verbs are mainly used for accessing, creating, updating and deleting resources. In particular, for the implementation of our REST API we have used the Java API for RESTful Web Services JAX-RS [39], as well as the support of Jersey [51]. JAX-RS is a standard and portable API that was designed to simplify the development of RESTful Web services and their clients in Java. Jersey RESTful Web Services is an open source framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs. Jersey is not only a JAX-RS Reference Implementation, but it also provides its own API that extends the JAX-RS toolkit with additional features and utilities in order to make the RESTful service and client

development easier. The REST API is deployed in an Apache Tomcat [68] version 8, which is a widely known open source software which implements a container for servlets.

Concerning the database, we have used MySQL Standard Edition [52] to keep all the data in our server machine. MySQL provides a reliable and efficient open source database, as well as MySQL WorkBench, which is an integrated development, design and administration environment.

As for the message broker that will allow the reception of events in our SOA 2.0, we have opted for RabbitMQ [73]. RabbitMQ is a lightweight, scalable and flexible open source message broker, which has a large community and team that produce additional clients, features, plugins, manuals, et cetera. RabbitMQ runs on many operating systems and cloud environments and can be deployed using clustered and federated models. One of the key features of Rabbit MQ is its support for multiple messaging protocols, such as AMQP, MQTT, and STOMP. Besides, RabbitMQ offers a management UI that makes it user friendly and facilitates the tracing of the implemented system. Among the communication protocols, we have chosen the standard and open protocol AMQP 0.9.1 [50], which is natively supported by RabbitMQ.

As an ESB that allows us to integrate our services and events into a SOA 2.0, we have chosen Mule ESB [49]. Mule is a lightweight and scalable Java-based ESB and integration platform that facilitates the connection among several applications as well as data exchange among them. It enables the interaction between applications transparently, regardless of the operating system where they are deployed, the technologies used for their implementation as well as the transport protocol used for the communications. It offers several services for routing, security, transaction management, message transformation, and so on and so forth. It also provides a user-friendly graphical design environment called Anypoint Studio and they recently offered the integration that allows developers to Manage APIs, Deployments and so on as Anypoint Framework.

As a CEP engine we have chosen Esper [21]. Esper is a recognized and open-source Java-based software engine for CEP. Esper can rapidly process large volumes of incoming data. For this purpose, Esper provides Event Processing Language (EPL). EPL extends SQL standard and permits precise definition of the patterns to be detected. The Esper compiler compiles EPL into byte code in a JAR file for its deployment. The Esper runtime loads and executes such byte code.

Esper does real-time streaming data processing, using parallelization and multithreading when necessary and it is highly scalable. Besides, it provides the option of implementing distributed stream processing along several machines as well as horizontal scalability, should it be necessary. According to their documentation with Esper version 8.1.0, performance reaches about 7.1 million events per second [36].

Finally, we have selected Android [32] as the operating system for app's development because it is the operating system with the highest market share right now: Android's market share compared to other operating systems is 74.25% (August 2020 [66]). Android is an open source mobile operating system, based on the Linux Kernel. Besides, we have used SQLite [64] to store the personal data in the smartphone. SQLite is an open source and cross-platform SQL database engine; it is small, fast, self-contained and highly reliable, making it very suitable for use in mobile applications. For the notifications from the server side to the app we have used Firebase Cloud Message [25] due to its reliability and low resource consumption.

6.3 Implementation of the proposed architecture

Figure 3. represents the chosen technologies for the architecture previously presented in Fig. 1. This combination enables real-time context-aware data processing and notification; the communications, represented in Fig. 4, are as follows:

1. Pollen and symptoms data reach the ESB through the message queues: on the one hand, as shown in arrow (a1) pollen level data are submitted to one of the incoming queues. On the other, represented as arrow (a2) each time a user reports a symptom, the mobile application will send it to a message queue; from there, the rest of the process is transparent to the user. The submitted report will be composed of the location and the reported symptom, together with information on the pollen types to which the user is allergic.

2. The message queue will deliver the data to the ESB, where some transformations will take place so that their format can be processed by the CEP engine (arrow (b1)) and the relevant symptoms are stored in the statistical system (arrow (b2)).
3. The CEP engine will receive the data and check if the established pattern is fulfilled. The defined pattern meets the following conditions: frequent symptoms must be detected, so a minimum number of events must be reported at a given time in a given location. Specifically, for testing purposes we will send a notification if 5 occurrences of the same symptom are reported in the same location over a period of 5 h. After such time, the data are discarded and the process starts again. That is, if 4 dyspnoea symptoms are reported in Cádiz and then one dermatitis report is received for the same city, they should not be added up and trigger a notification, being different symptoms. Analogously, if the same symptom is reported 5 times but not in the same city, it should not be notified. In the future, an expert on allergies might establish more appropriate thresholds for notifications.
4. When the conditions are met, a complex event is generated and fed back to the ESB again (arrow (c)). This complex event is composed of the location and symptom attributes.
5. At this point, the system would have detected a frequent symptom in a location, so a notification could be sent to all users from the said location. However, the detected symptom may not affect all users in the city, but those who are allergic to a particular

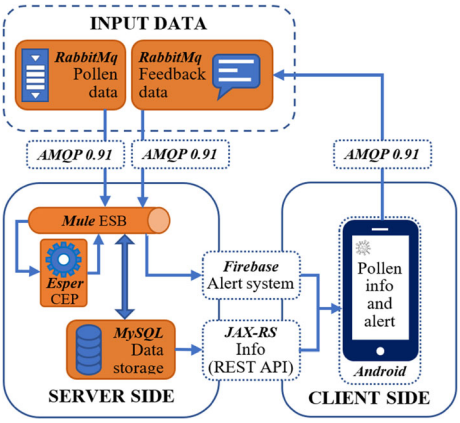


Fig. 3 Technical implementation of the architecture

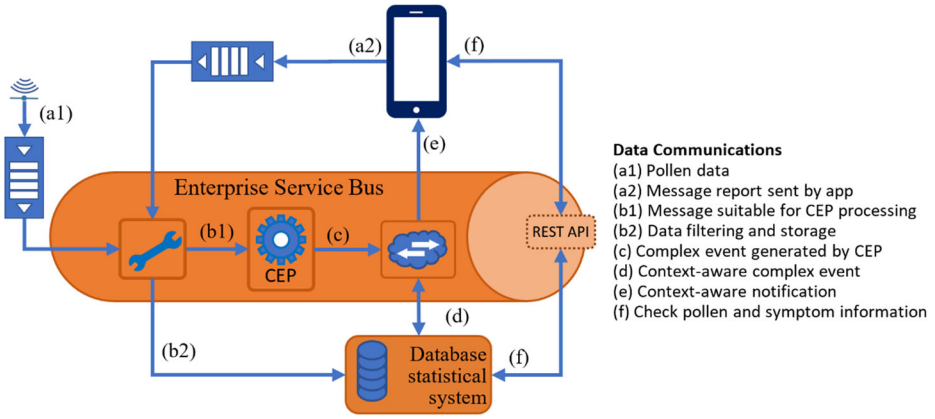
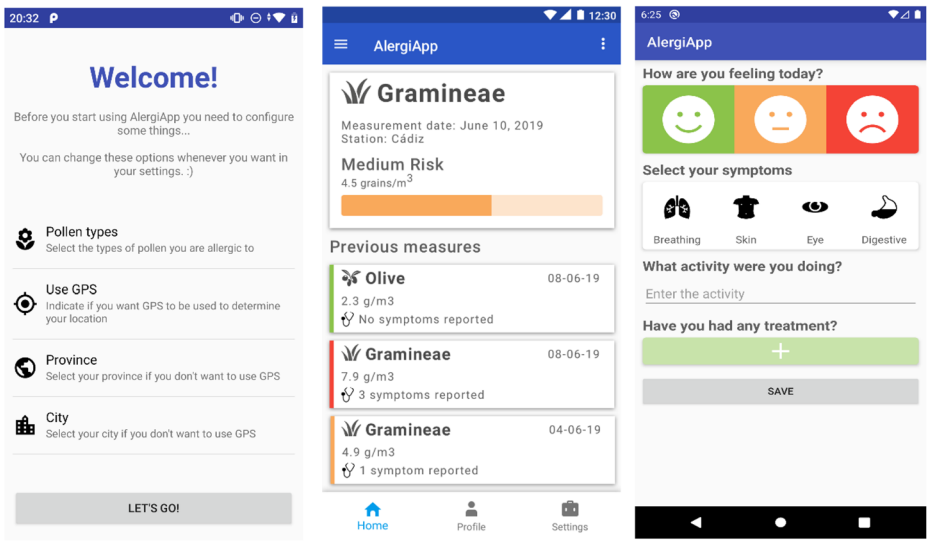


Fig. 4 Communication flows in the architecture

type of pollen. Therefore, to limit notifications to interested users, we will make a query to the statistical information system, which will return a list of pollen types that may have caused the detected symptom (arrow (d)) in order to submit the notifications only to those users allergic to the pollen type in question.

6. Therefore, the last step is to submit the notifications to subscribed users according to the complex event data (symptom and city) and the list of pollen types causing it (arrow (e)).
7. At any time, the mobile app accesses the REST service to check the current count of each pollen type, related symptoms as well as statistic information (arrow (f)).

In Fig. 5 we can see several screenshots of the implemented app: AlergiApp. In Fig. 5a) we can see the initial app screen, where we can indicate the pollen types which we are allergic to and if we want to activate the GPS or use a fixed location /for notification purposes. Figure 5b)



(a) (b) (c)

Fig. 5 AlergiApp Screenshots. **a** Welcome screen, **b** Current pollen level, **c** Symptoms report

shows the pollen concentration levels for a user who has indicated he/she is allergic to gramineae and olive. Finally Fig. 5c) shows the screen where the user can report how he/she is feeling and which symptoms he/she is experiencing today.

6.4 Security implementation in the proposed architecture

It is important to emphasize that we have secured the whole proposed system and all the communications in it, as explained below.

With respect to the API REST, we have implemented the following security features [62]:

- HTTPS is used for all connections.
- Access control is gained by means of username and password for the operation of reporting new symptoms and by means of API Keys for the rest of operations. Please bear in mind that the Tomcat server manages user access control through Digest access authentication.
- All entries are validated by using regular expressions.
- The input JSON is validated so that, in case of error, no error trace is returned, but an appropriate message understandable by the user is sent.
- All service management operations are protected by JSON Web Tokens.
- The server version is hidden in all error messages.
- We log the operations performed on the API, so that we can see how the API is being used at any time.

Regarding communications with RabbitMQ:

- TLS [69] is used to protect all information sent through the queue.

With respect to the mobile application:

- The communications of the app, both with the REST API and with the message queue, are always encrypted thanks to the use, as previously mentioned, of HTTPS and TLS, respectively.
- The app code is audited with SonarQube [63].
- The app code is obfuscated with ProGuard [33].

Finally, in relation to the database:

- The appropriate users, functions and privileges have been created to prevent unauthorized access to the database
- The database is audited using the following logs: general —connections and requests—, binary —modifications made to the database— and slow query —requests that take longer to execute—

It is important to emphasize that the user's privacy and personal data are fully protected. Firstly, this is because the user's personal data (what they are allergic to and their location for notifications) is stored locally on their device and therefore only they have access to it. Secondly, when the app sends the user's symptoms, location and physical activity to the

server via the REST API, such information is not associated to a particular user, but to a type of allergy. At no time is the person suffering from these symptoms communicated or stored in the server. The server sends all notifications to Firebase and then Firebase filters which user receives which notifications based on the topic to which the user's app subscribes. For example, if a user with an allergy reports an itchy eye symptom, the server receives the message that a user with a grass allergy has an itchy eye, without identifying the user. When the server considers that it should send a notification because the itchy eye threshold for grass has been exceeded, the server sends it to Firebase, and Firebase only sends it to those users whose app has subscribed to the grass topic. To sum up, the fact that no information about the user's identity is sent or stored in the server, together with all the security measures previously explained in terms of code and communications, makes the application totally secure in terms of user privacy.

Last but not least, to avoid the behaviour of a malicious user being able to seriously influence the statistics and therefore the notifications of symptoms detected to other users, we have limited the report that each user can make of each symptom to a daily one. This prevents the malicious user from repeatedly sending a symptom and significantly influencing the statistics and causing the threshold for notifications to be easily exceeded, as well as preventing a non-malicious user who lacks experience in using apps from sending the symptom many times in a short period of time, thus producing the same falsified result.

7 Experimental results

In this section, on the one hand, we have evaluated the system's performance and the resources consumed by AlergiApp; on the other hand, user satisfaction has been evaluated.

7.1 Performance and resource consumption evaluation

Given the great relevance of the system's performance as well as the low consumption the app should have, we have evaluated both parameters in our proposal.

First of all, we have tested the system's server-side performance, which may be impacted by a large number of users reporting symptoms at the same time. For such an evaluation, the system's server-side response time was evaluated when submitting an increasingly large number of symptoms reported by users. For this purpose, we used nITROGEN, an Internet of Things RandOm GENreator [26], in order to send a large number of random user reported symptoms to the system message entrance queue. Then, we measured the time required by the system to process such an amount of messages. Such processing includes receiving the messages from the queue, transforming them into the appropriate format to be processed by the CEP engine, processing them by the CEP engine and storing the result. We had the emulator, the queue and the rest of the system in three different machines: an Intel core i5 3.1 GHz with 8GB of RAM, an Intel core i5 3.6 GHz with 8GB RAM and an Intel core i7 3.5 GHz with 8GB RAM, respectively.

As we can see in Table 5, the processing time remains reasonable (less than 30 s) when processing up to 50,000 messages in a regular desktop machine, however, we do not expect to have 50,000 users reporting a symptom at the same time. Besides, the system's architecture is easily scalable and can be deployed in a more powerful server machine or a cloud one.

Table 5 Processing time with an increasing number of user reports

Reports per second	Server-side processing time (s)
1000	1.26
10,000	7.86
20,000	13.12
30,000	18.05
40,000	24.39
50,000	29.98

On the other hand, it is very important to keep the mobile app's resource consumption low, otherwise users will not be keen on using it. This is why we evaluated resource consumption for the mobile device. For this purpose, we used a BQ Aquaris V with Qualcomm Snapdragon 435 (octa-core at 1.4 GHz) processor, 4G of RAM memory and 3100 mAh battery with Android 7.1.2 (Nougat).

According to Berrocal et al. (see Table 2 in [13]), the consumption of data and battery for receiving a notification should be 407 bytes and 18.36 μAh, respectively. In our case, we kept the app running in isolation for a period of 8 h. In that period, other users reported up to 32 symptoms and the app received 5 notifications. The battery consumed by the mobile during such a period was 70 mA, less than 1% of the mobile's full battery, and only 20% of it was caused by mobile applications and 6% due to awakenings. Concerning data traffic, the app consumed 2440 bytes of data due to received notifications. The results are in line with the estimation given in [13]: 18.36 μAh for each notification would mean 91.8 μAh battery consumption for 5 notifications, and 407 bytes for each notification would mean 2035 bytes for 5 notifications. Besides, reporting a symptom of around 400 bytes consumes 5.1 Kb, which is also in line with the estimation provided by [13]: if posting 140 bytes implies 1.16 kb of data consumed, posting 400 bytes should be around 3.3kB consumption. Bear in mind that final values might depend on the protocols used and message headers needed for the communications. Therefore, we can conclude that battery and data consumptions remain reasonable and within the usual margins of an app's consumption patterns.

7.2 User evaluation

It is also of great importance to be aware of the app's usability and to know whether users are satisfied by the functionalities offered. For this purpose, we followed the user evaluation explained in this section.

7.2.1 The questionnaire

The questionnaire contained three sections: (i) participant demographics (age, gender and postcode) and technological competence (rated from 1 to 5), (ii) app usability and (iii) app utility and symptom management. These questions were not numbered in the questionnaire.

According to Sauro et al. [60], one of the most widely used standardized usability questionnaires for assessment of the perception of usability is the System Usability Scale (SUS) [40]. The SUS is a questionnaire with ten items, as shown below, which can be rated from 1 to 5 (from strongly disagree to strongly agree, respectively). The odd-numbered items have a positive tone, while even-numbered ones show negative appreciations. Accordingly, we have used the 10 questions in the SUS questionnaire in order to evaluate the app's usability:

1. I think I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Finally, the questions used to evaluate the app utility and how the user manage his/her symptoms are shown below:

11. When the allergy season strikes...

- I get information about pollen levels from other existent resources so I can act before they affect me.
- They always end up affecting me because I cannot predict them.

12. The most interesting features of AlergiApp are:

- Information about pollen levels.
- Frequent symptoms alerts from users with the same allergies as me.
- Alerts about increasing pollen concentrations in my area.
- Information about each pollen type.
- Allowing me to keep track of my allergy outbreaks.

13. For future improvements, you consider it critical…

- To include information about pollen from more cities.
- To include more types of pollen.
- To include a section with graphs of the most reported symptoms, alerts received, pollen levels, etc.
- To include more symptoms and treatments.
- Generation of PDF reports about the reported symptoms.
- To add an option to filter alerts and symptoms based on different criteria.

14. Currently I do not control my symptoms because there is no tool that allows it in a simple way. (Choose from 1 to 5, with 1 being completely disagreeing and 5 completely agreeing.)
15. Having up to date information on pollen levels can make me improve my quality of life, as I would be able to prevent or reduce an allergy attack. (Choose from 1 to 5, with 1 being completely disagreeing and 5 completely agreeing.)
16. I believe that notifications of frequent symptoms thanks to the collaboration of other citizens in my city are useful, as they may allow me to anticipate the manifestation of

such symptoms. (Choose from 1 to 5, with 1 being completely disagreeing and 5 completely agreeing.)

17. AlergiApp is useful to better understand how each type of pollen affects me and therefore to prevent uncomfortable symptoms. (Choose from 1 to 5, with 1 being completely disagreeing and 5 completely agreeing.)

7.2.2 Circulation of the questionnaire

We have decided the minimum size of the sample for the questionnaire following formulae (1), where Z is the level of confidence, p is the probability of success or expected prevalence, and d is the precision. We have set precision to 5%, probability of success to 95% and level of confidence to 1.44 (85%). This would imply 40 required participants in the questionnaire.

$$n = \frac{Z^2 * p * (1-p)}{d^2} \tag{1}$$

We managed to involve 42 people in the testing of the app and completion of the questionnaire. Therefore, 42 adults —17 to 68 years old—installed and tested AlergiApp for 2 weeks and, after that period of time they completed the questionnaire. Most participants were people known to the paper authors, since it was not easy to find and involve people with pollen allergies who were committed to the experiment. We also made a call for volunteers through some University of Cadiz virtual classrooms.

7.2.3 Results and analysis

According to participants' responses to the question on technological competence (*How competent are you with new technologies? (Choose from 1 to 5, 1 being the least experienced and 5 being a very competent user*)), participants had a large range of technological abilities, as shown in Table 6.

Concerning the usability of the app, we have calculated the SUS index of each survey respondent's answer. From this data, and taking into account the number of respondents (42) and the level of confidence set for the survey (1.44, which is 85%), we have calculated the data shown in Table 7, where we can see the mean of the SUS index, its standard deviation, the upper and lower limits of the Confidence Interval (CI) and the rating given based on the mean of the SUS (see ratings from Table 8.5 in [60]). We also calculated a two-factor rating taking into account eight items for subscale *Usable* and the two-item subscale *Learnable*, as explained in [60].

As we can see in Table 7, we got good results for the usability of the app, obtaining an A+ grade for SUS index mean as well as for the *Usable* and *Learnable* features.

In relation to the app's functionality and user symptom management, the results obtained from question 11 show that most of the users (88.1%) are affected by allergy because they

Table 6 Technological skills stated by survey respondents

Question 1 options	1	2	3	4	5
Number of respondents	3	2	5	11	21

Table 7 SUS rating for AlergiApp

SUS Rating	SUS Index mean	Standard Deviation	CI Upper Limit	CI Lower Limit	Mean Grade
AlergiApp	95.8	7.06	97.4	94.26	A+
Usable	95.9	6.71	94.41	97.40	A+
Learnable	95.5	10.63	93.17	95.54	A+

cannot predict it, as represented in Fig. 6a). This fact is mainly due to the lack of tools which allow allergy control in a simple way (see responses to question 14 in Fig. 6b)).

Figure 7 (question 12) shows that the feature in which users had more interest (31.2%) is the functionality of receiving alerts regarding frequent symptoms reported by other users, which is the main novelty of the proposed architecture and app. Additionally, other functionalities in which they also show more interest were having information about pollen (31.2%), followed by receiving alerts about pollen concentrations (10.4%).

As illustrated in Fig. 8, most participants (92.9%) totally agreed that AlergiApp is useful for a better understanding of how each pollen type affects them (question 17), whereas 90.5% totally agreed that notifications of frequent symptoms are useful to anticipate the manifestation of the mentioned symptoms (question 16). Note that knowing the increase of pollen levels and other citizens' symptoms in real time allows users to take preventing measures such as taking the medication if he/she is carrying it or avoiding physical activity outside, when possible. Finally, 97.6% totally agreed that having up to date information on pollen levels can make them improve their quality of life (question 15). Therefore, this clearly indicates that the application is very useful and fosters user fidelity, ensuring that the application fulfils its functionality successfully.

Finally, regarding future improvements (question 13), as shown in Fig. 9, the most demanded features are the inclusion of pollen information from more cities, including a section with graphs about the most reported symptoms, received alerts, pollen levels, etc., as well as the addition of more symptoms and treatments.

Table 8 depicts the mean, standard deviation and confidence interval, for 85% confidence level, for questions 14, 15, 16 and 17. As a conclusion, the survey results are very reliable.

8 Discussion

In this section, we analyse how our proposal has addressed the suggestions made by the pulmonologist regarding the usefulness of the app for patients with seasonal allergies, as well summarizing a comparison of results obtained with those from other related proposals.

If we review the features discussed in the meetings held with the pulmonologist, we can confidently affirm that:

Table 8 Statistical analysis for questions 14 to 17

	Question 14	Question 15	Question 16	Question 17
Mean	4.83	4.95	4.88	4.88
Standard Deviation	0.44	0.31	0.4	0.45
Confidence Interval	4.73–4.93	4.88–5	4.79–4.97	4.78–4.98

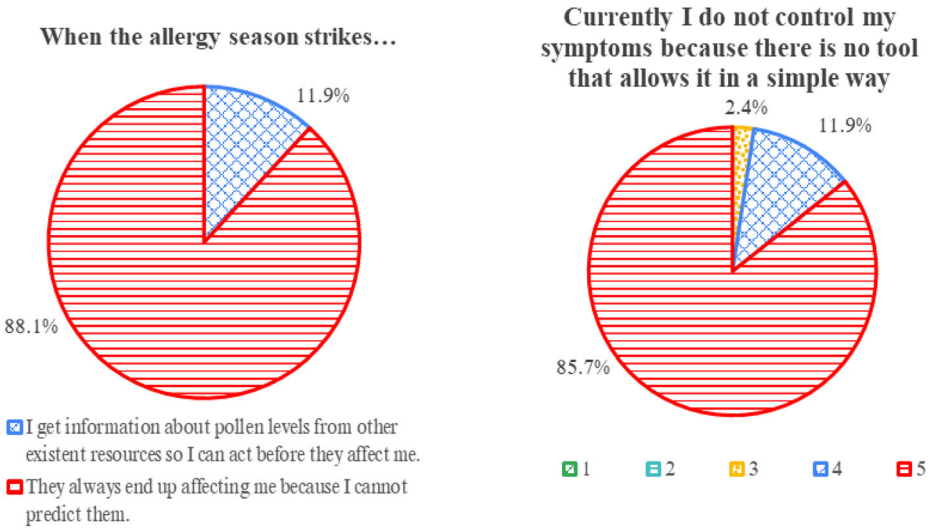


Fig. 6 Results to questions 11 and 14. **a** Responses to question 11, **b** Responses to question 14

- As shown in Table 7, we obtained very good results in regarding the app's usability, despite the fact that some of the participants declared to have low technological skills (see Table 6). This will permit the use of the app by many patients with other serious lung diseases such as COPD, which can be strongly aggravated by seasonal allergy episodes, and who are usually older people with limited technological ability.
- The inclusion of the physical activity feature (see Section 5.1) is not only beneficial to allergy sufferers in general, but, again, even more particularly valuable for people with other respiratory diseases such as COPD. This is one of the most worrying diseases at the respiratory level, which increasingly affects older people, along with other causes, due to our current society's tendency to sedentary life. Therefore, this type of patients usually has recommendations to perform daily physical exercise, to the extent of their abilities, so allergen alerts being adjusted when they are about to do physical exercise would be very positive for them, since such beneficial activity could turn against them if they were unaware of the fact that there is a high incidence of pollen on that particular day.

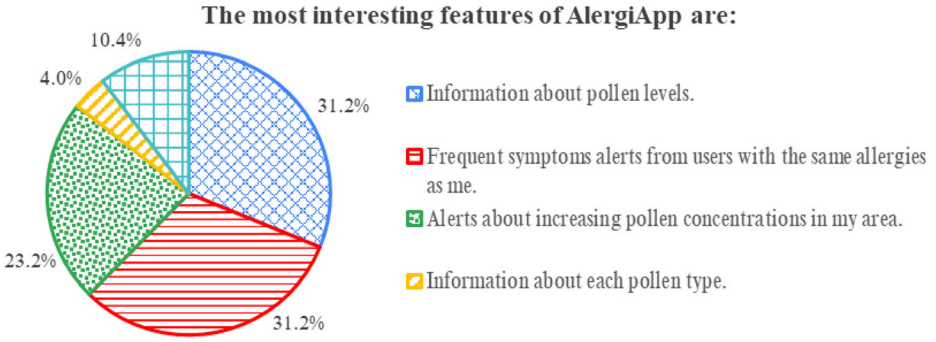


Fig. 7 Results to question 12

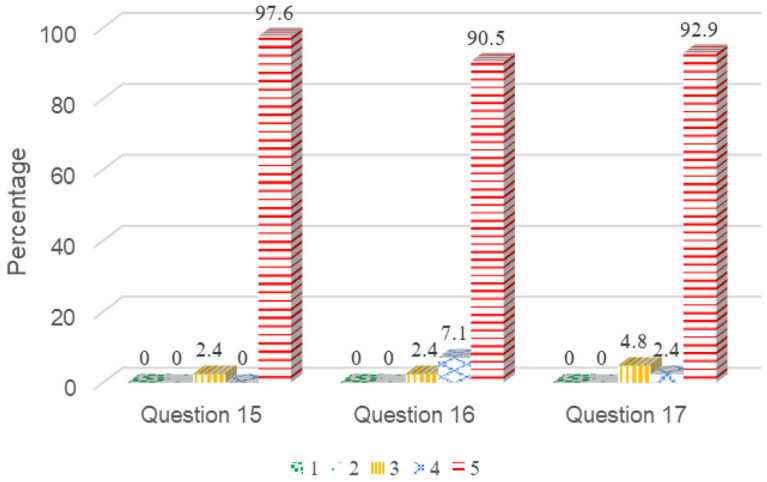


Fig. 8 Results to questions 15, 16 and 17

- On the other hand, as explained in Section 5.2, the information collected and stored consists of the reported symptoms and physical activity, the list of pollen types to which the user is allergic and the city in which the user is located when reporting them. This can be very useful to conduct statistics to be analysed by pulmonologists that could be used not only for research purposes, but also to improve the care of patients who come to the doctor's daily and give them information and suggestions on how to prevent and treat the most common symptoms that are being experienced that season.

Taking all these considerations into account, added to the benefits analysed through the survey to users in Section 7, we can say that the application and the proposed system are very useful.

When we try to compare our proposal with related works more specifically, we see that certain comparisons cannot be extended beyond what is reflected in Table 1. When looking at the specific frameworks and middlewares for context awareness [8, 9, 31, 44, 76], they mainly focus on context discovery and service adaptation. In our proposal, the discovery of the context is already provided by the architecture and application itself (location, user allergies, physical activity) and the alert

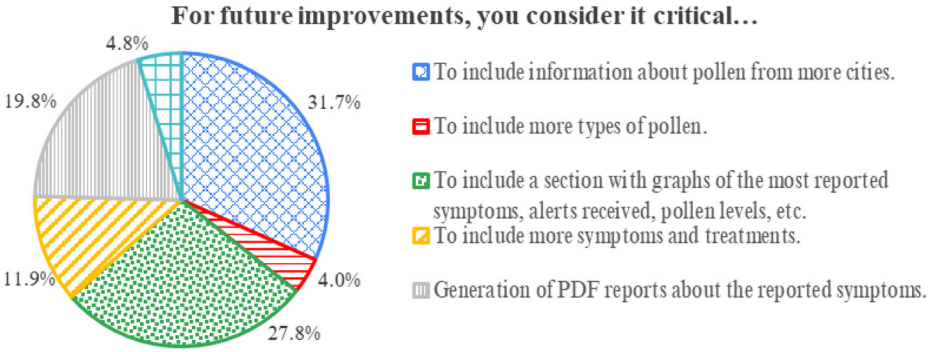


Fig. 9 Results to question 13

personalization is the service offered; however, we go one step further with regard to citizen collaboration and the personalization of notifications. The same applies to context-based recommendation systems, many of which are based on static contexts and the user's environment [4, 18, 78], but which do not take into account at all the contribution of the rest of the app's users, and above all the ability to process the whole context and adapt the service in real time. Collaboration has mainly been approached in research from the perspective of achieving a common goal in teams [12, 14, 16, 48, 70, 71], but not as a way of obtaining a greater understanding of a circumstance or context from which diverse individuals could benefit from an independent approach, and not with the aim of achieving a common end. Furthermore, although context-aware applications are beginning to emerge in the health field [15, 41, 58], the contextual information they take into account is limited to the user and again does not benefit from the collaboration and feedback provided by other users.

Table 9 shows a comparison of our proposal with other systems and applications for allergy control. The table evaluates the following features:

- Real Time: the application provides real-time information on pollen counts.
- Alerts: the application provides alerts when high concentrations of pollen are identified.
- Personalized Alerts: the application provides alerts when high concentration of pollens to which the user is allergic are found.
- Physical Activity: the application adapts the alerts to the physical activity the user is doing.
- Citizen Collaboration: the application benefits from the information provided by other app users.
- Statistics: the application benefits from the statistics obtained about the symptoms other users are experiencing, taking into account the pollen to which the reporting user has allergies.
- History: the app saves historical information about the symptoms the user suffered, for him/her to be able to follow the evolution of his/her symptoms.
- Suggestion: information about prospective allergens that may have caused the user a particular symptom on a particular date.
- Usability: app rated value in Google Play. Please note that the feature evaluation has been made based on the information available on Google Play.

Table 9 Comparison of AlergiApp to other allergy control applications and systems

	Real Time	Alerts	Person. Alerts	History	Phisical Activity	Citizen Collab.	Statistics	Sugg.	Usab.
[3]	Yes	No	No	No	No	No	No	No	3.93
[5]	Yes	Yes	Yes	Yes	No	No	No	No	2.1
[20]	Yes	No	No	No	No	No	No	No	4
[37]	Yes	No	No	No	No	No	No	No	3.2
[43]	Yes	Yes	Yes	No	No	No	No	No	4.2
[57]	Yes	No	No	Yes	No	No	No	No	4
[61]	Yes	Yes	Yes	Yes	No	Yes	Partially	Partially	3.5
[65]	Yes	No	No	No	No	No	No	No	2.5
[72]	Yes	No	No	Yes	No	Yes	Partially	No	4.8
[74]	Yes	Yes	Yes	Yes	No	No	No	Partially	3.4
AlergiApp	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	–

As we can see, all the applications provide *real-time* information on pollen counts, which is quite reasonable since it is usually the main functionality of this type of application; half of them provide *alerts*, which are usually *personalized* according to user allergies and location. It is also interesting to highlight the feature that allows users to keep a *history* of their symptoms. Also, half of the proposals offer a *history* of user's symptoms, as AlergiApp does, so that the user can observe his/her evolution. Currently, users can show this history to their attending pulmonologists so that they can better understand how the allergy affects that patient and provide better care.

On the other hand, as we can check, most of the apps cannot personalize alerts related to the *physical activity* being performed by the user. Besides, only 2 approaches benefit from citizen collaboration: in those apps, information from the user's symptoms is collected and the statistics are calculated, which permit sharing with other users the most common symptoms, however, there are two main differences from our proposal. First of all, the *statistics* are calculated with data from all users reporting symptoms but not taking into account which pollen the reporting user is allergic to, so other users cannot know if the symptom is related to to their own allergies. Secondly, the symptoms are shown in the app and no alert is sent to the users suffering the same allergies. Besides, our proposal submits alerts of people reporting the same symptom from a given threshold, we are not aware if the other apps use any threshold or if they show the symptom as soon as one person reports it. Furthermore, we go one step beyond in making use of the statistics: when the user can check his/her history of symptoms, he/she can also see a suggestion of which allergen may have caused such symptoms on that particular day and geographical area. Such a *suggestion* is based on previously provided statistics based on all user reports within the area and with such symptoms. This information can be of great benefit to the user since it helps him/her to find out about allergens that may be affecting him/her and that he/she was not aware of. This feature is only provided by our proposal, while others only support the user to find out what they may be allergic to by showing them which allergen they have been exposed to, such as [61, 74].

Therefore, if we were to highlight the great benefits of our context-aware and collaborative software architecture and app, we would point out the democratization of data through citizens' *collaboration* when reporting and alerting other users based on the former daily symptoms contextualized to their type of allergy, as well as the fact that alerts can be adapted to the contextualized *physical activity*; thirdly, our system also has the benefit of providing additional information about the allergen that may have caused the symptoms. All these are extremely useful features which could in turn be susceptible to other IoT and smart city domains.

Finally, concerning *usability*, we have included the apps' rates in Google Play as of January 11th, 2021; some of them have good rates, such as, ordering them from the highest to the lowest, [3, 43, 61, 72, 74]. For AlergiApp we have no such information since it is not provided in Google Play, but the usability questionnaire and other questions results show good satisfaction of respondents with the app (see Section 7.2.3).

9 Conclusions and future work

This paper has proposed a collaborative context-aware mobile application, supported by a software architecture integrated by technologies which permits collaboration from citizens in a simple way, and real-time processing and notification of the corresponding contextual alerts efficiently, as well as maintaining low resource consumption in the app. Such key features

guarantee the app's success, according to the survey carried out by 42 people of different ages and abilities with new technologies.

The impact of this paper is not limited to the proposed app but paves the way for the implementation of other context-aware collaborative apps that have considerable scope and demand in the field of the IoT and smart cities. The new proposals could use the approached architecture or adapt it to other specific needs, with the guarantee of having (1) high real-time processing capacity of the data provided by citizen collaboration, (2) an effective mechanism to adapt notifications to user context, and (3) low resource consumption in the app.

Besides, in the case of this particular scenario, being able to easily obtain detailed pollen information and real-time contextual alerts is also of special interest to healthcare personnel, as they will be able to advise patients on the basis of up-to-date information or to anticipate the symptoms that the citizens may present when they go to their general practitioner or even to the emergency services, as well as for research purposes. Furthermore, thanks to the feedback system, it will be possible to obtain statistical information that could be of interest for medical research or as a historical database to predict symptoms for the next allergy season. It could also be studied, for instance, how each pollen type affects the organism, since right now allergy symptoms do not distinguish between pollen types, even though there might be a relationship.

In our future work, we plan to use the proposed architecture in a case study where more additional contexts reported by the user can be taken into account: we will study whether there is a need to adapt the architecture to the new scenario and whether it remains within adequate margins of efficiency [53]. In addition to the actions that we have already adopted to avoid the behaviour of malicious or inexperienced users being able to influence the statistics and symptom notifications which are submitted, we will study what other options are appropriate to this end. Of course, we will also evaluate and attend the suggestions provided by survey respondents for future improvements, carry out further evaluations with a larger number of users and for longer periods of time and consider whether the app should be available in Google Play Store.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11042-021-10759-6.

Acknowledgements This work was supported by the Spanish Ministry of Science and Innovation and the European Union FEDER Funds [grant numbers RTI2018-093608-B-C33, RED2018-102654-T]. We would like to thank the personal support offered by Puerto Real Hospital pulmonologist Carmen Maza, as well as researcher Winfried Lamersdorf, for their interest in our ongoing research projects. We would also like to thank Rubén Rivas for his support with the initial versions of the server-side architecture.

References

1. Abowd GD et al. (1999) Towards a Better Understanding of Context and Context-Awareness. Presented at the 1st International Symposium on Handheld and Ubiquitous Computing , Karlsruhe, Germany. https://doi.org/10.1007/3-540-48157-5_29.
2. Aguilera U et al (2016) Collaboration-Centred Cities through Urban Apps Based on Open and User-Generated Data. Sensors 16(7):1022. https://doi.org/10.3390/s16071022
3. aha! Allergiezentrum Schweiz: Pollen-News (2020), https://play.google.com/store/apps/details?id=com.getunik.aha.pollen. Accessed 11 Jan 2021

4. Alhamid MF, Rawashdeh M, al Osman H, Hossain MS, el Saddik A (2015) Towards context-sensitive collaborative media recommender system. Multimed Tools Appl 74(24):11399–11428. https://doi.org/10.1007/s11042-014-2236-3
5. Almirall: Polen Control (2021), https://play.google.com/store/apps/details?id=com.almiralldiagnostics. Accessed 11 Jan 2021
6. American Academy of Allergy, Asthma & Immunology (AAAAI): Your Questions Answered on Air Pollution and Asthma | AAAAI (2018), https://www.aaaai.org/conditions-and-treatments/library/asthma-library/air-pollution-asthma, last accessed 2021/01/11.
7. Asthma and Allergy Foundation of America (aafa): Allergy Facts and Figures (2018), https://www.aafa.org/allergy-facts/, last accessed 2021/01/11.
8. Athanasopoulos D, Zarras AV, Issarny V, Pitoura E, Vassiliadis P (2008) CoWSAMI: Interface-aware context gathering in ambient intelligence environments. Pervasive Mob Comput 4(3):360–389. https://doi.org/10.1016/j.pmcj.2007.12.004
9. Baralis E, Cagliero L, Cerquitelli T, Garza P, Marchetti M (2010) CAS-mine: providing personalized services in context-aware applications by means of generalized rules. Knowl Inf Syst 28(2):283–310. https://doi.org/10.1007/s10115-010-0359-z
10. Basic JMS API Concepts - the Java EE 6 tutorial (2013), https://docs.oracle.com/javaee/6/tutorial/doc/bncdx.html, last accessed 2021/01/11
11. Behmann F, Wu K (2015) Collaborative internet of things (C-IoT): for future smart connected life and business. John Wiley and Sons, Inc, Hoboken
12. Benítez-Guerrero E, Mezura-Godoy C, Montané-Jiménez LG (2012) Context-aware Mobile collaborative systems: conceptual modeling and case study. Sensors. 12(12):13491–13507. https://doi.org/10.3390/s121013491
13. Berrocal J, Garcia-Alonso J, Vicente-Chicote C, Hernández J, Mikkonen T, Canal C, Murillo JM (2016) Early analysis of resource consumption patterns in mobile applications. Pervasive Mob Comput 35:32–50. https://doi.org/10.1016/j.pmcj.2016.06.011
14. Botev J et al. (2017) CollaTrEx – Collaborative Context-Aware Mobile Training and Exploration. In: Brooks, A.L. and Brooks, E. (eds.) Interactivity, Game Creation, Design, Learning, and Innovation. pp. 113–120 Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-55834-9_13.
15. Casino F et al (2018) Smart healthcare in the IoT era: a context-aware recommendation example. In: 2018 international symposium in sensing and instrumentation in IoT era (ISSI). Pp. 1–4. IEEE, Shanghai. https://doi.org/10.1109/ISSI.2018.8538106
16. Chung HM (2012) Toward implementing a mobile collaborative system. In: 2012 International Conference on Systems and Informatics (ICSAI2012). pp. 1248–1252. IEEE, Yantai, China. https://doi.org/10.1109/ICSAI.2012.6223262
17. De Backere F et al (2017) The OCarePlatform: a context-aware system to support independent living. Comput Methods Prog Biomed 140:111–120. https://doi.org/10.1016/j.cmpb.2016.11.008
18. De Pessemier T et al (2016) A user-centric evaluation of context-aware recommendations for a mobile news service. Multimed Tools Appl 75(6):3323–3351. https://doi.org/10.1007/s11042-014-2437-9
19. Dey AK (2001) Understanding and using context. Pers Ubiquit Comput 5(1):4–7. https://doi.org/10.1007/s007790170019
20. Dr. Safadi & Associates, Inc.: Allergy Pollen Count (2018), https://apps.apple.com/us/app/allergy-pollen-count/id903685327. Accessed 11 Jan 2021
21. EsperTech: Esper (2021), http://www.espertech.com/esper/. Accessed 11 Jan 2021
22. European Academy of Allergy and Clinical Inmunology (EAACY) (2015) Tackling the Allergy Crisis in Europe - Concerted Policy Action Needed
23. European Investment Bank, Deloitte: Horizon 2030: Looking ahead to challenges and opportunities (2019) , https://www.eib.org/attachments/strategies/horizon_2030_en.pdf
24. European Research Group in the Internet of Things: The Internet of Things 2012 New Horizons (2012), http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf, last accessed 2021/01/11
25. Firebase Cloud Messaging | Send notifications across platforms for free (2021), https://firebase.google.com/products/cloud-messaging. Accessed 11 Jan 2021
26. García-de-Prado, A. (2020) : nITROGEN: Internet of Things RandOm GENreator, https://ucase.uca.es/nITROGEN/. Accessed 11 Jan 2021
27. Garcia-de-Prado A, Ortiz G, Boubeta-Puig J (2017) COLLECT: COLLaborativE ConText-aware service oriented architecture for intelligent decision-making in the internet of things. Expert Syst Appl 85:231–248. https://doi.org/10.1016/j.eswa.2017.05.034

28. Garcia-de-Prado A et al (2017) CARED-SOA: a context-aware event-driven service-oriented architecture. IEEE Access 5:4646–4663. https://doi.org/10.1109/ACCESS.2017.2679338
29. Garcia-de-Prado A et al (2018) Air4People: a smart air quality monitoring and context-aware notification system. J Univ Comput Sci 24(7):846–863. https://doi.org/10.3217/jucs-024-07-0846
30. Gil D et al (2016) Internet of Things: A Review of Surveys Based on Context Aware Intelligent Services. Sensors 16(7):E1069. https://doi.org/10.3390/s16071069
31. Gilman E, Su X, Davidyuk O, Zhou J, Riekki J (2011) Perception framework for supporting development of context-aware web services. Int J Pervasive Comput Commun 7(4):339–364. https://doi.org/10.1108/17427371111189665
32. Google: What is Android? (2021), https://www.android.com/intl/en_uk/what-is-android/. Accessed 11 Jan 2021
33. Guardsquare: ProGuard (2021), https://www.guardsquare.com/en/products/proguard. Accessed 11 Jan 2021
34. Harchay A et al. (2015) A context-aware approach for personalized Mobile self-assessment. JUCS - J Univ Comput Sci 8. https://doi.org/10.3217/jucs-021-08-1061.
35. Immanuel VA, Raj P (2015) Enabling context-awareness: A service oriented architecture implementation for a hospital use case. Presented at the International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) , Davangere, India October. https://doi.org/10.1109/ICATCCT.2015.7456886.
36. Inc, E.: 7+ Million Events-Per-Second (2021), https://www.espertech.com/2019/03/07/6-million-events-per-second/. Accessed 11 Jan 2021
37. Innovatech Innovatech Informatica & Telecomunicaciones: Polen REA (2019), https://play.google.com/store/apps/details?id=com.innovatech.rea. Accessed 11 Jan 2021
38. Inzinger C, Hummer W, Satzger B, Leitner P, Dustdar S (2014) Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems: event-based monitoring and adaptation for distributed systems. Softw Pract Exp 44(7):805–822. https://doi.org/10.1002/spe.2254
39. Java.net: Java API for RESTful Services (JAX-RS) (2021), https://jax-rs-spec.java.net/. Accessed 11 Jan 2021
40. Jordan PW et al. (1996) Eds: SUS: a “quick and dirty” usability scale. In: Usability Evaluation In Industry. CRC Press. https://doi.org/10.1201/9781498710411.
41. Kim J, Lee D, Chung KY (2014) Item recommendation based on context-aware model for personalized u-healthcare service. Multimed Tools Appl 71(2):855–872. https://doi.org/10.1007/s11042-011-0920-0
42. Kim K et al (2016) i-RM: An intelligent risk management framework for context-aware ubiquitous cold chain logistics. Expert Syst Appl 46:463–473. https://doi.org/10.1016/j.eswa.2015.11.005
43. Kitakits: Alerta Polen (2021), https://play.google.com/store/apps/details?id=alerte.pollen. Accessed 11 Jan 2021
44. Li F et al. (2010) COPAL: an adaptive approach to context provisioning. Presented at the October. https://doi.org/10.1109/WIMOB.2010.5645051.
45. Luckham DC (2002) The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley, Reading, Massachusetts
46. Luckham DC (2012) Event processing for business: organizing the real-time enterprise. John Wiley & Sons, Hoboken, N.J
47. Mobile Vs Desktop Internet Usage Statistics (2020), https://saasscout.com/statistics/mobile-desktop-usage/, last accessed 2021/01/11
48. Montané-Jiménez LG et al (2014) Towards a Context-Aware Framework for Improving Collaboration of Users in Groupware Systems. EAI Endorsed Trans Context-Aware Syst Appl 1(1):e4. https://doi.org/10.4108/casa.1.1.e4
49. MuleSoft: Mule ESB (2021), http://www.mulesoft.org/. Accessed 11 Jan 2021
50. OASIS: AMQP is the Internet Protocol for Business Messaging | AMQP (2021), https://www.amqp.org/about/what. Accessed 11 Jan 2021
51. Oracle Corporation: JERSEY (2020). RESTful Web Services in Java, https://jersey.java.net/. Accessed 11 Jan 2021
52. Oracle Corporation: MySQL: MySQL Standard Edition (2021), https://www.mysql.com/products/standard/. Accessed 11 Jan 2021
53. Ortiz G, Garcia-de-Prado A, Berrocal J, Hernandez J (2019) Improving resource consumption in context-aware Mobile applications through alternative architectural styles. IEEE Access 7:65228–65250. https://doi.org/10.1109/ACCESS.2019.2918239
54. Papazoglou M (2012) Web services and SOA: principles and technology. Pearson Education, Essex, England ; New York

55. Papazoglou M, Heuvel WVD (2006) Service-oriented design and development methodology. Int J Web Eng Technol 2(4):412–442. https://doi.org/10.1504/IJWET.2006.010423
56. Peinado S, Ortiz G, Dodero JM (2015) A metamodel and taxonomy to facilitate context-aware service adaptation. Comput Electr Eng 44:262–279. https://doi.org/10.1016/j.compeleceng.2015.02.004
57. Pollen Sense LLC: Pollen Wise (2020), https://play.google.com/store/apps/details?id=com.PollenSense.PollenWise. Accessed 11 Jan 2021
58. Rahman MDA et al (2014) Context-aware multimedia services modeling: an e-Health perspective. Multimed Tools Appl 73(3):1147–1176. https://doi.org/10.1007/s11042-013-1595-5
59. Roy Thomas F (2000) Architectural styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine
60. Sauro J, Lewis JR (2016) Chapter 8 - standardized usability questionnaires. In: quantifying the user experience: practical statistics for user research. Pp. 185–248. Morgan Kaufmann, Cambridge
61. Screencode: Pollen (n.d.) , https://play.google.com/store/apps/details?id=screencode.pollenwarndienst, last accessed 2021/01/11
62. Siriwardena P (2019) Advanced API Security: OAuth 2.0 and Beyond. Apress
63. SonarSource S.A,: SonarQube (2021), https://www.sonarqube.org/developer-edition/index_emea.html. Accessed 11 Jan 2021
64. SQLite Consortium: SQLite Home Page (2021), https://www.sqlite.org/index.html. Accessed 11 Jan 2021
65. STARx Technology Corporation: AccuPollen™ Allergy Tracker (2020), https://play.google.com/store/apps/details?id=com.accupollen. Accessed 11 Jan 2021
66. StatCounter Global Stats: Mobile Operating System Market Share Worldwide (2021), https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed 11 Jan 2021
67. Sundermann CV, Domingues MA, Conrado MS, Rezende SO (2016) Privileged contextual information for context-aware recommender systems. Expert Syst Appl 57:139–158. https://doi.org/10.1016/j.eswa.2016.03.036
68. The Apache Software Foundation (2021) : Apache Tomcat® - Welcome!, http://tomcat.apache.org/. Accessed 11 Jan 2021
69. Thomas SA (2000) SSL & TLS essentials: securing the web. Wiley, New York
70. Truong H et al. (2007) Escape - an adaptive framework for managing and providing context information in emergency situations. Presented at the Second European Conference, EuroSSC, Kendal, England. https://doi.org/10.1007/978-3-540-75696-5_13.
71. Truong H-L et al. (2008) inContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms. In: International Symposium on Applications and the Internet. pp. 118–125, Turku, Finland. https://doi.org/10.1109/SAINT.2008.70.
72. University of Melbourne: Melbourne Pollen Count (2020), https://play.google.com/store/apps/details?id=com.plenum.pollen. Accessed 11 Jan 2021
73. VMware, In: Messaging that just works — RabbitMQ (2020), https://www.rabbitmq.com/. Accessed 11 Jan 2021
74. Wlab Ltd: Sensio Air, Pollen Pollut (2020), https://play.google.com/store/apps/details?id=com.sensioair.sensio. Accessed 11 Jan 2021
75. Xu Y, Yin J, Deng S, N. Xiong N, Huang J (2016) Context-aware QoS prediction for web service recommendation and selection. Expert Syst Appl 53:75–86. https://doi.org/10.1016/j.eswa.2016.01.010
76. Yu J, Han J, Sheng QZ, Gunarso SO (2012) PerCAS: an approach to enabling dynamic and personalized adaptation for context-aware services. In: Liu C et al (eds) Service-oriented computing, pp. 173–190 Springer. Berlin Heidelberg, Berlin, Heidelberg
77. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M (2014) Internet of things for smart cities. IEEE Internet Things J 1(1):22–32. https://doi.org/10.1109/JIOT.2014.2306328
78. Zavala L et al. (2011) Mobile, collaborative, context-aware systems. In: Proceedings of the 4th AAAI Conference on Activity Context Representation: Techniques and Languages, pp. 79–84 AAAI Press

Affiliations

Pablo Caballero [1] · Guadalupe Ortiz [1] · Alfonso Garcia-de-Prado [1] · Juan Boubeta-Puig [1]

Pablo Caballero
pablo.caballero@uca.es

Alfonso Garcia-de-Prado
alfonso.garciadeprado@uca.es

Juan Boubeta-Puig
juan.boubeta@uca.es

[1] UCASE Software Engineering Group, School of Engineering, University of Cádiz, Avda. de la Universidad de Cádiz 10, 11519 Puerto Real, Cádiz, Spain